Apple Assembly Line

$2.40

Volume 8 -- Issue 8 May, 1988

The Last Issue

With deep regret I have to inform you that this is the last issue of Apple Assembly Line. The income at S-C Software has suddenly and finally decreased beyond the point at which AAL can continue to be published. I have found employment elsewhere, and for the time being have put S-C Software to sleep. I will be working full time now as a programmer for a certain major manufacturer of Apple peripheral boards, for whom I have written a considerable amount of firmware over the last five years: Applied Engineering.

If your subscription expiration date is still in the future, then I owe you something in place of those future issues of the newsletter. I would like to repay you with materials I have on hand, such as back issues of the newsletter, copies of my software products, or perhaps books.

In order to determine how many months remain on your subscription, look at the mailing label. At the right end of the top line you will see a four-digit number, such as 8812. The first two digits are the year, the last two digits are the month of what should have been your last issue. The issue in your hands is issue 8805. Subtract 88 from the year of your expiration date, and multiply the remainder by 12; add the product to the month of your expiration date, and subtract 5; the result should be the number of issues I owe you. For example, if your expiration code is 8903, I owe you 10 more issues.

If possible, I would like to give you back issues in place of future ones. Please write me and let me know which issues you would like. Include some second choices in case I run out of some of the issues. If you had a subscription which included monthly disks, I will also include the disks with the back issues.

<< continued on next page >>

Of course, if you already have all of the back issues you will want to work out some other arrangement. Alternatively, you may want to select some item(s) from among the following software and hardware products and apply your remaining subscription toward the purchase price. The actual amount of your credit depends on how much you paid for your subscription; I'll trust you to work that out.

By the time you are reading this, I will be working during the day for my new employer. Please write with your request, or call and leave complete details on my answering machine at (214) 324-2050, by September 30, 1988.

S-C Macro Assembler Version 2.0 . . . . . DOS $100, ProDOS $100
. . both for $120
Version 2.0 DOS Upgrade Kit for 1.0/1.1/1.2 owners $20
ProDOS Upgrade Kit for Version 2.0 DOS owners $30

Cross Assemblers for owners of S-C Macro Assembler . . $32.50 to $50 each
(More info on these below)

S-C DisAssembler (ProDOS only) . . . . . without source code $30
with source code $50

ProVIEW (ProDOS-based disk utility program) $20

Full Screen Editor for S-C Macro (with complete source code) $49

S-C Cross Reference Utility . . . . . without source code $20
with source code $50

S-C Word Processor, both DOS & ProDOS, both 40- & 80-columns,
with complete source code . . . $50

DP18 and DPFP, double precision math for Applesoft,
including complete source code. . . $50

S-C Documentor (complete commented source code of Applesoft ROMs)
. . . . . $50

AAL Quarterly Disks . . . . . each $15, or any four for $45

Vinyl disk pages, 6"x8.5", hold two disks each . . . . . 10 for $6 *

Sider 20 Meg Hard Disk, includes controller & software . . . ($695) $550 +

65802 Microprocessor, 4 MHz (Western Design Center) . . . . . $25 *

quikLoader EPROM System (SCRG) . . . . . ($179) $170 *
PROmGRAMER (SCRG) . . . . . ($149.50) $140 *

* These items add $2 for first item, $.75 for each additional item for shipping in USA.
+ Add $10 for shipping in USA.
Customers outside USA inquire for postage needed.
Texas residents please add 8% sales tax to all orders.
<< Master Card, VISA, Discover and American Express >>

About S-C Macro Cross Assemblers.....Bob Sander-Cederlof

Combining the versatile Apple II with the S-C Macro Cross Assemblers provides a cost effective and powerful development system for many different microprocessors.

All of the S-C Macro Assemblers are all identical in operation: only the language assembled is different. Each S-C Macro Assembler is a complete macro assembler with an integrated, co-resident program editor, and operates in any member of the Apple II family having at least 48K RAM and one disk drive (ProDOS versions require 64K RAM). Each is written in 6502 assembly language for execution in Apple II series computers, but assembles standard mnemonics for the target processor into binary object code for that processor. The standard version assembles code for either 6502, normal 65C02, the Rockwell special version of the 65C02, or 65C816 microprocessors. Cross Assemblers are available for a wide variety of microprocessors.

S-C Cross Assemblers are sold as supplements to the standard S-C Macro Assembler. The S-C Macro Assembler, complete with 120-page reference manual, costs $100 for either the DOS 3.3 or ProDOS version, or $120 for both; once you have it, you may add as many Cross Assemblers as you wish at a much lower price. The following S-C Macro Cross Assembler versions are now available:

Microprocessor		DOS 3.3	ProDOS	Both
Motorola:	6800,1,2,8/6301	$50	$50	$70
	6805	$50	$50	$70
	6809	$32.50	n/a	
	68HC11	$50	$50	$70
	68000	$50	n/a	
Mitsubishi:	50740 series	$50	$50	$70
Intel:	8048 family	$32.50	n/a	
	8051 family	$32.50	n/a	
	8080/8085	$32.50	n/a	
Zilog:	Z-80	$32.50	n/a	
	Z-8	$32.50	n/a	
RCA:	1802/1805	$32.50	n/a	
DEC:	LSI-11	$50	n/a	
General Instruments:	GI-1650	$50	$50	$70
	GI-1670	$50	$50	$70
Sharp:	LH5801	$50	$50	$70

The assembled object code may be directed either to Apple memory or to a DOS 3.3 or ProDOS binary file. If you have an EPROM burner, the object can be burned into EPROMs. (We recommend and sell the SCRG PromGramer, which burns any 24- or 28-pin EPROM from 2716 through 27512. It fits in an Apple slot, and is only $140.) Other options are to download your object code to a target system via a serial port, to download to a ROM-emulator, or to execute direcly out of Apple memory with a co-processor card. Co-processor cards for the Z-80, 6809, and 68000 are available from various manufacturers.

This seems like a good time to give some sort of index to all of the pieces of AppleWorks I have discussed in the past five issues of AAL. The following table is in order by address within the code.

1000-1185	JMP vector, etc.	Mar 88, Feb 88
1186-119D	CALL.FOR.AWPROGRAM.DISK	Mar 88
119E-122C	LOAD.PROGRAM.SEGMENT.A	Mar 88
122D-1340	LOAD.SEGMENT.FROM.DISK	Mar 88
1341-1365	APPEND.STRINGS	Dec 87
1366-136D	CLEAR.MAIN.WINDOW	this issue
136E-139C	FLUSH.KEYBUF.CHECKING.ESC	this issue
139D-13B1	CLR.LINE.X.TO.LINE.Y	this issue
13B2-13C0	CHECK.KEYBUF	Feb 88
13C1-14CF	various messages	Mar 88, this issue
14D0	INVERSE.FLAG	Jan 88
14D1-153F	DISPLAY.STRING	Jan 88
1540-1543	FUN.CLR.LINE	Jan 88
1544-1549	FUN.CLR.CH.TO.EOL	Jan 88
154A-1551	FUN.HOME	Jan 88
1552-1575	FUN.CLR.CH.TO.EOS	Jan 88
1576-1592	FUN.GOTO.XY	Jan 88
1593-1598	FUN.CURSOR.LEFT	Jan 88
1599-159B	FUN.HANG.UP	Jan 88
159C-15A0	more of CURSOR.LEFT	Jan 88
15A1-15AA	FUN.CURSOR.RIGHT	Jan 88
15AB-15BB	FUN.CURSOR.UP	Jan 88
15BC-15C9	FUN.CURSOR.DOWN	Jan 88
15CA-160A	SCROLL	Jan 88
160B-160E	FUN.INVERSE	Jan 88
160F-1616	FUN.NORMAL	Jan 88
1617-1621	FUN.CORNER.BR	Jan 88
1622-1628	FUN.CURSOR.BOL	Jan 88
1629-1633	FUN.CORNER.TL	Jan 88
1634-1644	FUN.FULL.SCREEN	Jan 88
1645-165D	FUN.BEEP	Jan 88
165E-1715	FUN.SHUFFLE	Jan 88
1716-1737	BASE.CALC	Jan 88
1738-1778	CLR.CH.TO.EOL	Jan 88
1779-179C	FUNTBL	Jan 88
179D-17D0	CONVERT.A.TO.RJBF.STRING	this issue
17D1-1814	DIVIDE.P0.BY.P2	this issue
1815-1817	Another HANG.UP	Jan 88
1818-1822	BEEP.AND.CLEAR.KEYBUF	this issue
1823-1836	MOVE.CURSOR.TO.XY	Feb 88
1837-1841	SHOW.HELP.STRING	this issue
1842-184F	WARN.IF.FULLDESK	this issue
1850-186B	SHOW.FULLDESK.WARNING	this issue
186C-1871	SHOW.COMMAND.ENTRY	this issue
1872-1879	CALL.ORGANIZER	this issue
187A-18AC	COPY.SCRN.LINE.TO.0900	Feb 88
18AD-18DC	GET.x.PARMS	Dec 87
18DD-18E3	GET.MENU.TABLE.INDEX	Apr 88
18E4-18F1	MAKE.MENU.LINE.NORMAL	Apr 88
18F2-190B	MAKE.MENU.LINE.INVERSE	Apr 88
190C-191C	RESTORE.ESCAPE.AND.HELP	Apr 88
191D-1A73	SELECT.MENU.LINE	Apr 88
1A74-1AFB	REVERSE.A.SCREEN.LINE	Apr 88

```
1AFC-1AFF  SET.PRODOS.BITMAP              Mar 88
1B00-1B0A  CLR.PRODOS.BITMAP              Mar 88
1B0B-1B2A  DRAW.TOP.AND.BOTTOM.LINES      this issue
1B2B-1B33  POST.CHANGE.FLAG               this issue
1B34-1B4D  MULTIPLY.X.BY.Y                this issue
1B4E-1B83  MULTIPLY.P0.BY.P2              this issue
1B84-1BAB  MOVE.BLOCK.DOWN                Dec 87
1BAC-1BDE  MOVE.BLOCK.UP                  this issue
1BDF-1BF0  POP.ESCAPE.ROAD.MAP            this issue
1BF1-1C13  WAIT.FOR.SPACE.RETURN.OR.ESCAPE  this issue
1C14-1C20  variables for PRINTER          this issue
1C21-1D0D  PRINTER.DRIVER                 this issue
1D0E           ???
1D0F-1D2F  PUSH.ESCAPE.ROAD.MAP           this issue
1D30-1D34  various variables              Feb 88
1D35-1D45  AW.KEYIN                       Feb 88
1D46-1DDA  KEYIN.ANOTHER.CHAR             Feb 88
1DDB-1E7F  KEYIN.ANALYSIS                 Feb 88
1E80-1E89  MOVE.CURSOR.TO.TCOL.TROW       this issue
1E8A-1E93  SAVE.GOTO.XY                   this issue
1E94-1EA8  MAP.SCRN.CHARS.TO.INTERNAL     Feb 88
1EA9-1EB3  POINT.PSTR.AT.0A00             Feb 88
1EB4-1EBE  MAP.LOWER.TO.UPPER             this issue
1EBF-1ED8  FILTER.LC.TO.UC                Dec 87
1ED9-1EF7  COMPARE.STRINGS                Dec 87
1EF8-1F09  MOVE.STRING                    Dec 87
1F0A-1F3D  READ.KEYBOARD                  Feb 88
1F3E-1F9F  DISPLAY.AT                     this issue
1FA0-1FA6  TRUNCATE.TO.79.IF.OVER.80      this issue
1FA7-1FD0  POLL.KEYBOARD                  Dec 87
1FD1-1FDF  DELAY.TENTHS                   Jan 88
1FE0-1FE8  CLEAR.KEYBUF                   Feb 88
1FE9-1FF4  DISPLAY.TOKEN.X                this issue
1FF5-2028  DISPLAY.ON.LINE.23             this issue
2029-2092  DISPLAY.MENU.LINE               Apr 88
```

As you can see, I am trying to fill in all the gaps this month. There is still more code beyond $2092 in the main section of AppleWorks, running all the way up to $2E84, but I have not finished disassembling all of it yet.

DON LANCASTER STUFF

INTRODUCTION TO POSTSCRIPT

A 65 min user group VHS video with Don Lancaster sharing many of his laser publishing and Postscript programming secrets.

Includes curve tracing, $5 toner refilling, the full Kroy Kolor details, page layouts, plus bunches more.

$39.50

ASK THE GURU

An entire set of reprints to Don Lancaster's ASK THE GURU columns, all the way back to column one. Edited and updated.

Both Apple and desktop publishing resources are included that are not to be found elsewhere.

$24.50

APPLE IIc/IIe ABSOLUTE RESET

Now gain absolute control over your Apple! You stop any program at any time.

Eliminates all dropouts on your HIRES screen dumps. Gets rid of all hole blasting. For any IIc or IIe.

$19.50

POSTSCRIPT SHOW & TELL

Unique graphics and text routines the others don't even dream of. For most any Postscript printer.

Fully open, unlocked, and easily adaptable to your own needs. Available for Apple, PC, Mac, ST, many others.

$39.50

FREE VOICE HELPLINE

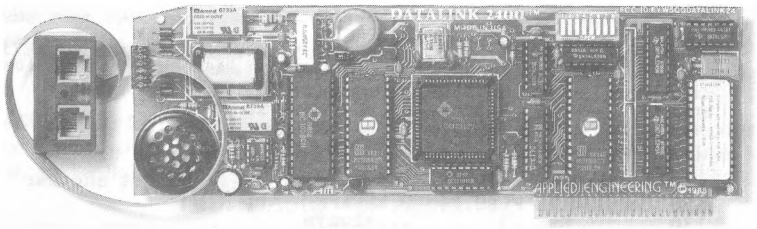
VISA/MC

SYNERGETICS

Box 809-SC Thatcher, AZ 85552 (602) 428-4073

```
                1020 *--------------------------------
109D-           1030 RJBF.STRING          .EQ $109D   Mar 88 issue
11A1-           1040 LOAD.PROGRAM.SEGMENT.A .EQ $11A1  Mar 88 issue
13B2-           1050 CHECK.KEYBUF         .EQ $13B2   Feb 88 issue
14D1-           1060 DISPLAY.STRING       .EQ $14D1   Jan 88 issue
1823-           1070 MOVE.CURSOR.TO.XY    .EQ $1823   Feb 88 issue
18AE-           1080 GET.4.PARMS          .EQ $18AE   Dec 87 issue
18B2-           1090 GET.2.PARMS          .EQ $18B2   Dec 87 issue
18B4-           1100 GET.A.PARMS          .EQ $18B4   Dec 87 issue
1B84-           1110 MOVE.BLOCK.DOWN      .EQ $1B84   Dec 87 issue
1D35-           1120 AW.KEYIN             .EQ $1D35   Feb 88 issue
1EA9-           1130 POINT.PSTR.AT.0A00   .EQ $1EA9   Feb 88 issue
1EF8-           1140 MOVE.STRING          .EQ $1EF8   Dec 87 issue
1FE0-           1150 CLEAR.KEYBUF         .EQ $1FE0   Feb 88 issue
2093-           1160 DISPLAY.STRING.P0    .EQ $2093
20AE-           1170 DRAW.BOTTOM.LINE     .EQ $20AE
20BE-           1180 REPEAT.CHAR.Y.X.TIMES .EQ $20BE
C082-           1190 ROM.D000             .EQ $C082
C083-           1200 RAM.D000             .EQ $C083
D002-           1210 MEASURE.FREE.MEMORY  .EQ $D002
D026-           1220 X.D026               .EQ $D026
D044-           1230 DISPLAY.K.AVAIL      .EQ $D044
D029-           1240 DISPLAY.FUNCTION.AND.ESCAPE.MAP .EQ $D029
FDED-           1250 MON.COUT             .EQ $FDED
                1260 *--------------------------------
24-             1270 MON.CH .EQ $24
25-             1280 MON.CV .EQ $25
36-             1290 CSWL   .EQ $36
37-             1300 CSWH   .EQ $37
80-             1310 PSTR   .EQ $80,81
88-             1320 FLAG.FOUND.ESCAPE    .EQ $88
89-             1330 Z.89   .EQ $89
                1340 #---used by Multiply & Divide----
91-             1350 M.REG          .EQ $91,92
93-             1360 MUL.X.FACTOR   .EQ $93
94-             1370 MUL.Y.FACTOR   .EQ $94
                1380 #--------------------------------
95-             1390 WINDOW.TOP     .EQ $95     initially 2  (3rd from top)
96-             1400 WINDOW.BOTTOM  .EQ $96     initially 21 (3rd from bottom)
98-             1410 PNTR           .EQ $98,99
9A-             1420 P0             .EQ $9A
9B-             1430 P1             .EQ $9B
9C-             1440 P2             .EQ $9C
9D-             1450 P3             .EQ $9D
9E-             1460 P4             .EQ $9E
9F-             1470 P5             .EQ $9F
A0-             1480 COUNT          .EQ $A0,A1
A1-             1490 Z.A1           .EQ $A1
F0-             1500 DIVISOR        .EQ $F0,F1
F2-             1510 REMAINDER      .EQ $F2,F3
                1520 *--------------------------------
0A00-           1530 STR.A                .EQ $0A00
0C6C-           1540 FLAG.CURRENT.FILE .EQ $0C6C
0CC6-           1550 FLAG.FOUND.SPACE  .EQ $0CC6
0CE8-           1560 ESCAPE.ROAD.MAP   .EQ $0CE8  nine 21-byte entries
0EA8-           1570 OPEN.PRINTER.FLAG .EQ $0EA8
0EAD-           1580 TCOL   .EQ $0EAD
0EAE-           1590 TROW   .EQ $0EAE
0EB2-           1600 X.0EB2 .EQ $0EB2
0EB3-           1610 X.0EB3 .EQ $0EB3
0EB4-           1620 X.0EB4 .EQ $0EB4
                1630 *--------------------------------
                1640 *   Following 164 bytes are copy of first 164 bytes
                1650 *      in SEG.PR file (0F19...0FBD):
                1660 *--------------------------------
0F3D-           1670 CURRENT.PRINTER.NO .EQ $0F3D    Currently Active Printer # (1-3)
                1680 *--------------------------------
                1690 *   Three 36-byte data areas, one for each printer
                1700 *      1:  $F51-F74
                1710 *      2:  $F75-F98
                1720 *      3:  $F99-FBC
                1730 *--------------------------------
0F61-           1740 X.0F61 .EQ $0F61
0F70-           1750 X.0F70 .EQ $0F70
0F71-           1760 X.0F71 .EQ $0F71
0F73-           1770 X.0F73 .EQ $0F73
                1780 *--------------------------------
0FC4-           1790 FLAG.DONT.ANALYZE.KEY  .EQ $0FC4
0FC6-           1800 SLOTROM.MAP            .EQ $0FC6 ...FCD  (8 BYTES)
0FEF-           1810 FLAG.APPLE.2C          .EQ $0FEF  01=Apple //c, else 00.
0FF3-           1820 HIDE.FULLDESK.WARNING  .EQ $0FF3
0FF4-           1830 CHAR.FOR.BOTTOM.LINE   .EQ $0FF4
0FF5-           1840 KBYTES.DESKTOP.LEFT    .EQ $0FF5,FF6
                1850 *--------------------------------
```

The new DataLink™ 2400 modem from Applied Engineering, it's a lot more than just twice as fast.



Applied Engineering's new DataLink™ 2400. Simply put, the finest modem on the market for your Apple IIgs, IIe or II+.

Bring home a world of information . . . from up to the minute flight information to whole libraries of resource materials. Even download free software and games.

Twice the speed.

At transmission speeds up to 2400 bps (bits-per-second), Applied Engineering's new DataLink 2400 is capable of putting text on the screen faster than the human eye can follow. That means you can capture a great deal more material in less time than with 1200 bps modems. And *unlike other modems*, the DataLink 2400 comes complete with powerful, easy-to-use communications software.

Complete communications software included.

Both our new DataLink 2400 and our DataLink 1200 modems feature AE's exclusive communications software —on disk and in ROM—everything needed to get you immediately up and running. Our powerful DataTerm software for the IIgs and IIe supports VT-52 screen emulation, macros, file transfers, on-line time display, recording buffer and more. It even stores hundreds of phone numbers for auto-dialing and log on. And for II+ and 64K IIe owners, our OnLine 64 software has many of the same powerful features.

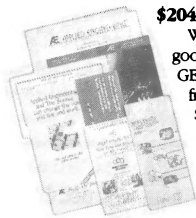
Worldwide compatibility.

The DataLink 2400 is fully compatible with Bell 103 and 212 protocols, as well as European protocol CCITT V.22 BIS, V.22 and V.21. It operates at varying transmission speeds from 0-300, 1200 and 2400 bps.

The new 2400, like our best-selling DataLink™ 1200, carries a full five year warranty and comes complete with two modular phone jacks for data and voice calls, a thoughtful feature that means fewer wires to connect. We also include an extra long telephone cable, in case your computer is across the room from your telephone jack. You can track the progress of calls either electronically or via on-board speaker. And built-in diagnostics reliably check transmission accuracy.

Packed with important features:

- Non-volatile memory for modem configuration
- Full Hayes AT compatibility
- Point-to-Point, ASCII Express, Access II compatibility, in addition to AE's included DataTerm and OnLine 64 software.
- Super Serial Card "Front End" for highest software compatibility (unlike others)
- Adaptive equalization and descrambling
- Hardware configuration for DSR and DCD
- PC Transporter (MS-DOS) compatibility
- FCC certified design



$204.90 in freebies.

We also throw in a nice collection of goodies—a free subscription to the GEnie network worth $29.95, $60 of free on-line time from NewsNet, a free $50 subscription to the Official Airline Guide and a fee-waived membership to The Source worth $49.95 plus $15 of free on-line time.

That's $204.90 worth of free memberships, discounts and on-line time when you purchase the powerful DataLink 2400 at **$239.**

DataLink 1200 reduced.

Loaded with all the features of the new 2400, (except CCITT, DSR/DCD and non-volatile ROM configurations) our 1200 bps DataLink modem, complete with software and freebies, is an affordable alternative at only **$179.**

DataLink 1200.....$179
DataLink 2400.....$239

Order today!

To order or for more information, see your dealer or call (214) 241-6060 today, 9 am to 11 pm, 7 days. Or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 7% sales tax. Add $10 outside U.S.A.

AE APPLIED ENGINEERING™
The Apple enhancement experts.
(214) 241-6060
P.O. Box 5100, Carrollton, TX 75011

Prices subject to change without notice. Brands and product names are registered trademarks of their respective holders.

```
                        1860          .ph $1366
                        1870 *--------------------------------
                        1880 *    CLEAR WINDOW, THEN SET FULL SCREEN
                        1890 *--------------------------------
                        1900 * (1366) 100C 25E2 2710 2ACA 2D34
                        1910 CLEAR.MAIN.WINDOW
1366- A6 95             1920          LDX WINDOW.TOP
1368- A4 96             1930          LDY WINDOW.BOTTOM
136A- 20 9D 13          1940          JSR CLR.LINE.X.TO.LINE.Y
136D- 60                1950          RTS
                        1960 *--------------------------------
                        1970 *    Flush buffer, but set flags if <SPC> or <ESC> found
                        1980 *--------------------------------
                        1990 * (136E) 100F
                        2000 FLUSH.KEYBUF.CHECKING.ESCAPE.AND.SPACE
136E- A9 00             2010          LDA #0
1370- 85 88             2020          STA FLAG.FOUND.ESCAPE
1372- 8D C6 0C          2030          STA FLAG.FOUND.SPACE
1375- A9 01             2040          LDA #1          Signal not to analyze char in KEYIN
1377- 8D C4 0F          2050          STA FLAG.DONT.ANALYZE.KEY
137A- 20 B2 13          2060 .1       JSR CHECK.KEYBUF
137D- F0 18             2070          BEQ .3          ...no chars in buffer
137F- 20 35 1D          2080          JSR AW.KEYIN
1382- C9 1B             2090          CMP #$1B        <ESC>?
1384- D0 06             2100          BNE .2          ...no
1386- A9 01             2110          LDA #1          ...yes, set flag and get another
1388- 85 88             2120          STA FLAG.FOUND.ESCAPE
138A- D0 EE             2130          BNE .1          ...always
138C- C9 20             2140 .2       CMP #$20        <SPACE>?
138E- D0 EA             2150          BNE .1          ...no, get another
1390- A9 01             2160          LDA #1          ...yes, set flag and get another
1392- 8D C6 0C          2170          STA FLAG.FOUND.SPACE
1395- D0 E3             2180          BNE .1          ...always
1397- A9 00             2190 .3       LDA #0          now buffer is empty
1399- 8D C4 0F          2200          STA FLAG.DONT.ANALYZE.KEY   Restore char-analysis in KEYIN
139C- 60                2210          RTS             return
                        2220 *--------------------------------
                        2230 *    Clear screen from line (X) to line (Y), and
                        2240 *       set window to full screen.
                        2250 *       (X)=top line to clear
                        2260 *       (Y)=bottom line to clear
                        2270 *--------------------------------
                        2280 * (139D) 1012 136A
                        2290 CLR.LINE.X.TO.LINE.Y
139D- 8E AF 13          2300          STX CLRSTR+4
13A0- A2 4F             2310          LDX #79
13A2- 20 23 18          2320          JSR MOVE.CURSOR.TO.XY
13A5- 20 93 20          2330          JSR DISPLAY.STRING.PO
13A8- AB 13             2340          .DA CLRSTR
13AA- 60                2350          RTS
                        2360 *--------------------------------
13AB- 06                2370 CLRSTR   .DA #6           6 bytes in string
13AC- 0C                2380          .HS 0C          Set bottom-right corner of window
13AD- 05 00 00          2390          .HS 05.00.00 Go to xx,yy
13B0- 04                2400          .HS 04          Clear inside window
13B1- 0F                2410          .HS 0F          Back to a full-scrn window
                        2420 *--------------------------------
                        2430          .ph $13C1
                        2440 *--------------------------------
                        2450          .MA MSG         MACRO TO SHORTEN LISTING
                        2460          .DA #:1-*-1
                        2470          .AS "]1"
                        2480 :1
                        2490          .EM
                        2500          .MA AS          MACRO TO SHORTEN LISTING
                        2510          .AS /]1/
                        2520          .EM
                        2530 *--------------------------------
13C1-                   2540 MSG..1   >MSG "Place the AppleWorks PROGRAM disk in Drive 1 and
1402- 0C                2550 MSG..2   .DA #MSG..3-*-1
1403- 81                2560          .HS 81          Open-Apple picture          press Return.  "
1404-                   2570          >AS "-? for Help"
140F- 1E                2580 MSG..3   .DA #MSG..4-*-1
1410-                   2590          >AS "Type entry or use "
1422- 81                2600          .HS 81          Open-Apple picture
1423-                   2610          >AS " commands  "
142E-                   2620 MSG..4   >MSG "Press Space Bar to continue  "
144C-                   2630 MSG..5   >MSG "Do you really want to do this"
146A-                   2640 MSG..6   >MSG "Type number, or use arrows, then press Return  "
149A- 35                2650 MSG..7   .DA #MSG..X-*-1
149B- 0A                2660          .HS 0A          inverse
149C-                   2670          >AS " WARNING.  Desktop is full.  Action not completed. "
14CF- 0B                2680          .HS 0B          normal
                        2690 MSG..X
```


May, 1988.....Copyright (C) S-C SOFTWARE

SPECIAL !!! EXPANDED RAM/ROM BOARD: $39.00

Similar to our $30 RAM/ROM dev board described below. Except this board has two sockets to hold your choice of 2-2K RAM, 2-2K ROM or even 2-4K ROM for a total of 8K. Mix RAM and ROM too. Although Apple limits access to only 2K at a time, soft switches provide convenient socket selection. Hard switches control defaults.

IMPROVED]|[IN A MAC (ver 2.0): $75.00

Now includes faster graphics, UniDisk support and more! Bi-directional data transfers are a snap! This Apple II emulator runs DOS 3.3/PRODOS (including 6502 machine language routines) on a 512K MAC or MACPLUS. All Apple II features are supported such as HI/LO-RES graphics, 40/80 column text, language card and joystick. Also included: clock, RAM disk, keyboard buffer, on-screen HELP, access to the desk accessories and support for 4 logical disk drives. Includes 2 MAC diskettes (with emulation, communications and utility software, plus DOS 3.3 and PRODOS system masters, including Applesoft and Integer BASIC) and 1 Apple II diskette.

SCREEN.GEN: $35.00

Develop HI-RES screens for the Apple II on a Macintosh. Use MACPAINT (or any other application) on the MAC to create your Apple II screen. Then use SCREEN.GEN to transfer directly from the MAC to an Apple II (with SuperSerial card) or IIc. Includes Apple II diskette with transfer software plus fully commented SOURCE code.

MIDI-MAGIC for Apple //c: $49.00

Compatible with any MIDI equipped music keyboard, synthesizer, organ or piano. Package includes a MIDI-out cable (plugs directly into modem port - no modifications required!) and 6-song demo diskette. Large selection of digitized QRS player-piano music available for 19.00 per diskette (write for catalog). MIDI-MAGIC compatible with Apple II family using Passport MIDI card (or our own input/output card w/drum sync for only $99.00).

FONT DOWNLOADER & EDITOR: $39.00

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Use with any Word Processor program capable of sending ESC and control codes to printer. Switch back and forth easily between standard and custom fonts. Special functions (like expanded, compressed etc.) supported. Includes HIRES screen editor to create custom fonts and special graphics symbols. For Apple II, II+, //e. Specify printer: Apple Imagewriter,Apple Dot Matrix, C.Itoh 8510A (Prowriter), Epson FX 80/85, or Okidata 92/192.

*** FONT LIBRARY DISKETTE #1: $19.00** contains lots of user-contributed fonts for all printers supported by the Font Downloader & Editor. Specify printer with order.

DISASM 2.2e : $30.00 ($50.00 with SOURCE Code)

Use this intelligent disassembler to investigate the inner workings of Apple II machine language programs. DISASM converts machine code into meaningful, symbolic source compatible with S-C, LISA, ToolKit and other assemblers. Handles data tables, displaced object code & even provides label substitution. Address-based triple cross reference generator included. DISASM is an invaluable machine language learning aid to both novice & expert alike. Don Lancaster says DISASM is "absolutely essential" in his ASSEMBLY COOKBOOK.

The 'PERFORMER' CARD: $39.00 ($59.00 with SOURCE Code)

Converts a 'dumb' parallel printer I/F card into a 'smart' one. Simple command menu. Features include perforation skip, auto page numbering with date & title, large HIRES graphics & text screen dumps. Specify printer. MX-80 with Graftrax-80, MX-100, MX-80/100 with Graftraxplus, NEC 8092A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93.

'MIRROR' ROM: $25.00 ($45.00 with SOURCE Code)

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Basic modes: Dumb Terminal, Remote Console & Programmable Modem. Features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back, printer buffer, 80 col card & shift key mod support.

RAM/ROM DEVELOPMENT BOARD: $30.00

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip (6116 type RAM for program development or 2716 EPROM to keep your favorite routines on-line). Maps into $Cn00-CnFF and $C800-CFFF.

C-PRINT For The APPLE //c: $69.00

Connect standard parallel printers to an Apple //c serial port. Separate P/S included. Just plug in and print!

Unless otherwise specified, all Apple II diskettes are standard (not copy protected!) 3.3 DOS.

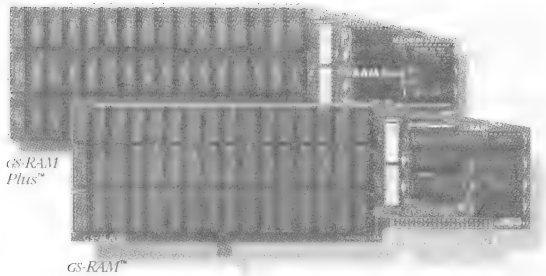
Avoid a $3.00 handling charge by enclosing full payment with order. VISA/MC and COD phone orders OK.

RAK-WARE 41 Ralph Road W. Orange N J 07052 (201) 325-1885

```
                    2700 *-------------------------------
                    2710         .ph $179D
                    2720 *-------------------------------
                    2730 * (179D) 1018 2037 25A0 25C5 25D6
                    2740 *   Convert (A) to right-justified, blank-filled
                    2750 *      decimal string in RJBF.STRING.  Also return
                    2760 *      the three digits in A,X,Y.  In both places,
                    2770 *      the three digits are in ASCII, with leading
                    2780 *      zeroes converted to blanks.
                    2790 *-------------------------------
                    2800 CONVERT.A.TO.RJBF.STRING
179D- A0 30         2810         LDY #'0'       Start with ASCII zero
179F- C9 64         2820 .1      CMP #100       Count the hundreds
17A1- 90 06         2830         BCC .2         ...none left
17A3- 38            2840         SEC            take 100 out
17A4- E9 64         2850         SBC #100
17A6- C8            2860         INY            and count it
17A7- D0 F6         2870         BNE .1         ...always
17A9- A2 30         2880 .2      LDX #'0'       Start with ASCII zero for 10's
17AB- C9 0A         2890 .3      CMP #10        Count the tens
17AD- 90 06         2900         BCC .4         ...none left
17AF- 38            2910         SEC            take 10 out
17B0- E9 0A         2920         SBC #10
17B2- E8            2930         INX            and count it
17B3- D0 F6         2940         BNE .3         ...always
17B5- 09 30         2950 .4      ORA #'0'       Change units to ASCII
17B7- C0 30         2960         CPY #'0'       Change leading zero to blank
17B9- D0 08         2970         BNE .5         ...not a leading zero
17BB- A0 20         2980         LDY #' '       ...lz, change to blank
17BD- E0 30         2990         CPX #'0'       Might be another lz, tens digit
17BF- D0 02         3000         BNE .5         ...not a leading zero
17C1- A2 20         3010         LDX #' '       ...lz, change tens to blank
17C3- 8C 9E 10      3020 .5      STY RJBF.STRING+1   store the 3 characters
17C6- A8            3030         TAY
17C7- 8E 9F 10      3040         STX RJBF.STRING+2
17CA- 8C A0 10      3050         STY RJBF.STRING+3
17CD- AD 9E 10      3060         LDA RJBF.STRING+1    also return in AXY
17D0- 60            3070         RTS
                    3080 *-------------------------------
                    3090 * (17D1) 101E
                    3100 DIVIDE.P0.BY.P2
17D1- 20 AE 18      3110         JSR GET.4.PARMS
17D4- B1 9A         3120 .1      LDA (P0),Y   Get two arguments via P0-P3
17D6- 99 91 00      3130         STA M.REG,Y  Dividend
17D9- B1 9C         3140         LDA (P2),Y
17DB- 99 F0 00      3150         STA DIVISOR,Y   Divisor
17DE- C8            3160         INY
17DF- C0 01         3170         CPY #1
17E1- F0 F1         3180         BEQ .1
17E3- A5 F0         3190         LDA DIVISOR
17E5- 05 F1         3200         ORA DIVISOR+1
17E7- D0 03         3210         BNE .2
17E9- 38            3220         SEC            Division by zero, return .CS.
17EA- B0 28         3230         BCS .5         ...always
17EC- A9 00         3240 .2      LDA #$00
17EE- 85 F2         3250         STA REMAINDER
17F0- 85 F3         3260         STA REMAINDER+1
17F2- A2 10         3270         LDX #$10
17F4- 26 91         3280 .3      ROL M.REG
17F6- 26 92         3290         ROL M.REG+1
17F8- 26 F2         3300         ROL REMAINDER
17FA- 26 F3         3310         ROL REMAINDER+1
17FC- 38            3320         SEC
17FD- A5 F2         3330         LDA REMAINDER
17FF- E5 F0         3340         SBC DIVISOR
1801- A8            3350         TAY
1802- A5 F3         3360         LDA REMAINDER+1
1804- E5 F1         3370         SBC DIVISOR+1
1806- 90 04         3380         BCC .4
1808- 84 F2         3390         STY REMAINDER
180A- 85 F3         3400         STA REMAINDER+1
180C- CA            3410 .4      DEX
180D- D0 E5         3420         BNE .3
180F- 26 91         3430         ROL M.REG
1811- 26 92         3440         ROL M.REG+1
1813- 18            3450         CLC
1814- 60            3460 .5      RTS
                    3470 *-------------------------------
                    3480 * (1815) 101B 1599 1815 2D45
1815- 4C 15 18      3490 HANG    JMP HANG
                    3500 *-------------------------------
```

For Those Who Want the Most. From Those Who Make the Best. GS-RAM.™

Now expand the IIGS' RAM and ROM with up to 8 MEG of "Instant On" memory with the all new GS-RAM!



GS-RAM has an all new design. A design that delivers higher performance including increased speed, greater expandability, and improved software.

More Sophisticated, Yet Easier to Use

GS-RAM works with all IIGS software. In fact *any* program that runs on Apple's smaller memory card runs on the GS-RAM. But with GS-RAM you can have more memory, improved performance, and almost unlimited expansion capabilities. We've designed the new GS-RAM to be easier to use too—you don't have to adjust the size of your RAM disk every time you use a DMA device. No other RAM card with more than 4 banks of memory installed can make the same claim.

More than Just Hardware

Each GS-RAM and GS-RAM Plus includes the most powerful set of IIGS software enhancements available anywhere. In fact, our nearest competitor offers only a fraction of the invaluable programs that we include with each GS-RAM card. This software includes the most powerful disk-caching program available, the GS-RAM Cache. The Cache will make most of your applications run up to 7 *times faster.* Also included is a diagnostic utility that lets you test your GS-RAM by graphically showing the location of any bad or improperly installed RAM chips. And for AppleWorks users, we give you our exclusive Expander program that dramatically enhances both the capabilities and speed of AppleWorks.

Making AppleWorks Even Better

Applied Engineering's Expander program eliminates AppleWorks internal memory limits allowing it to recognize up to 8 megabytes of desktop workspace. You can increase the limits from only 7,250 lines to 22,600 lines in the word processor and from 6,350 records to 22,600 records in the database. The Expander allows all of AppleWorks, including print functions, to automatically load into RAM. The clipboard size will increase from 255 to 2,042 lines maximum. GS-RAM will automatically segment larger files so you can save them onto multiple floppies. And GS-RAM provides a built-in print buffer that allows you to continue working in AppleWorks while your printer is still processing text. You can even load Pinpoint or MacroWorks and your favorite spelling checker into RAM for instant response.

Grow by Kilobytes or Megabytes

We offer GS-RAM in two configurations so you can increase your memory 256K at a time (GS-RAM) or a megabyte at a time (GS-RAM Plus). Both are IIGS compatible and both come with our powerful enhancement software. GS-RAM can hold up to 1.5 MEG of 256K chips and GS-RAM Plus can hold up to 6 MEG using 1 MEG chips. And since both use standard RAM chips (not high-priced SIMM's), you'll find expanding your GS-RAM or GS-RAM Plus easy, convenient, and very economical. For further expansion, you can plug a 2 MEG "piggyback" card into the GS-RAM's expansion port for up to 3.5 MEG of total capacity. Or up to a whopping 8 MEG on GS-RAM Plus. If a GS-RAM owner outgrows 3.5 MEG, he can easily upgrade to GS-RAM Plus for a nominal charge.

Permanent Storage for an "Instant On" Apple

With our RamKeeper™ back-up option, your GS-RAM or GS-RAM Plus will retain both programs and data while your IIGS is turned off! Now when you turn your IIGS back on, your favorite software is on your screen in under 4 seconds! With RamKeeper you can divide your IIGS memory into part "electronic hard disk," and part extended RAM. Even change the memory boundaries at any time—and in any way—you want. Because Applied Engineering has the most experience in the industry with battery-backed memory for the Apple, you are assured of the most reliable memory back-up system available. And in the world of battery-backed memory, *Reliability* is everything. That's why Applied Engineering uses state-of-the-art "GEL-CELL's" instead of Ni-Cad batteries (if Ni-Cads aren't discharged periodically, they lose much of their capacity). RamKeeper has about 6 hours of "total power failure" back-up time. That's 6 times the amount of other systems. But with power from your wall outlet, RamKeeper will back-up GS-RAM, GS-RAM Plus, or most other IIGS memory cards indefinitely. Should you ever have a "total power failure," RamKeeper switches to its 6-hour battery. When power returns, RamKeeper will automatically recharge the battery to full power. RamKeeper incorporates a dual-rate charger, status L.E.D.'s, and advanced power reducing circuitry. RamKeeper comes *complete* with battery, software, and documentation.

"In quality, performance, compatibility, expandability and support, Applied Engineering's GSRAM and GSRAM Plus are number one."

Steve Wozniak, the creator of Apple Computer

GS-RAM's Got it ALL!

- 5-year warranty — parts & labor
- 6 RAM banks (most cards have 4)
- Memory expansion port
- ROM expansion port
- Ultra-fast disk caching on ProDOS 8 *AND* ProDOS 16.
- Expands AppleWorks internal limits
- Includes hi-res self test
- No soldered-in RAM chips
- Expandable to 8 MEG
- No configuration blocks to set
- RamKeeper back-up option allows permanent storage of programs & data
- 15-day money-back guarantee
- Proudly made in the U.S.A.

GS-RAM with 256K	**$189**
GS-RAM with 512K	**$259**
GS-RAM with 1 MEG	**$399**
GS-RAM with 1.5 MEG	**$539**
GS-RAM with 2.5 to 3.5 MEG	**CALL**
GS-RAM Plus with 1-8 MEG	**CALL**
RamKeeper Option	**$179**

Order today!

See your dealer or call Applied Engineering today, 9 a.m. to 11 p.m. 7 days. Or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 7% sales tax. Add $10.00 outside U.S.A.

APPLIED ENGINEERING™

The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

Prices subject to change without notice

GS-RAM, GS-RAM Plus and RamKeeper are trademarks of Applied Engineering. Other brands and product names are registered trademarks of their respective holders.

```
                        3510 * (1818) 1021 1986 1A2D 1C0C 26CC
                        3520 BEEP.AND.CLEAR.KEYBUF
1818- A2 10             3530         LDX #$10      BEEP TOKEN
181A- 20 E9 1F          3540         JSR DISPLAY.TOKEN.X
181D- 20 E0 1F          3550         JSR CLEAR.KEYBUF
1820- A9 01             3560         LDA #$01
1822- 60                3570         RTS
                        3580 *-------------------------------
                        3590         .ph $1837
                        3600 *-------------------------------
                        3610 * (1837) 1027 201C
                        3620 SHOW.HELP.STRING
1837- 20 02 D0          3630         JSR MEASURE.FREE.MEMORY
183A- 20 3E 1F          3640         JSR DISPLAY.AT
183D- 42 17 02
1840- 14                3650         .DA #$42,#$17,MSG..2   "Apple-? for Help"
1841- 60                3660         RTS
                        3670 *-------------------------------
                        3680 WARN.IF.FULLDESK
1842- AD F5 0F          3690         LDA KBYTES.DESKTOP.LEFT
1845- 0D F6 0F          3700         ORA KBYTES.DESKTOP.LEFT+1
1848- D0 05             3710         BNE .1
184A- 20 50 18          3720         JSR SHOW.FULLDESK.WARNING
184D- A9 00             3730         LDA #$00
184F- 60                3740 .1      RTS
                        3750 *-------------------------------
                        3760 * (1850) 102D 184A
                        3770 SHOW.FULLDESK.WARNING
1850- AD F3 0F          3780         LDA HIDE.FULLDESK.WARNING
1853- D0 16             3790         BNE .2        Don't display anything
1855- A6 96             3800         LDX WINDOW.BOTTOM
1857- E8                3810         INX
1858- 8E 5F 18          3820         STX .1        Store line number to print on
185B- 20 3E 1F          3830         JSR DISPLAY.AT
185E- FF                3840         .DA #$FF
185F- 00 9A 14          3850 .1      .DA #$00,MSG..7  "WARNING"
1862- 20 F1 1B          3860         JSR WAIT.FOR.SPACE.RETURN.OR.ESCAPE
1865- AD F4 0F          3870         LDA CHAR.FOR.BOTTOM.LINE
1868- 20 AE 20          3880         JSR DRAW.BOTTOM.LINE
186B- 60                3890 .2      RTS
                        3900 *-------------------------------
                        3910 * (186C) 1030
                        3920 SHOW.ENTER.COMMAND.MSG
186C- 20 F5 1F          3930         JSR DISPLAY.ON.LINE.23
186F- 0F 14             3940         .DA MSG..3   "Type Entry or use Apple-commands"
1871- 60                3950         RTS
                        3960 *-------------------------------
                        3970 * (1872) 1033 238F
                        3980 CALL.THE.ORGANIZER
1872- A2 F8             3990         LDX #$F8
1874- 9A                4000         TXS
1875- A9 20             4010         LDA #$20
1877- 20 A1 11          4020         JSR LOAD.PROGRAM.SEGMENT.A
                        4030 *---Even though the above is a JSR, it never returns---
                        4040 *---from LOAD.PROGRAM.SEGMENT.A, because it enters-----
                        4050 *---the ORGANIZER directly-----------------------------
                        4060 *-------------------------------
                        4070         .ph $1B0B
                        4080 *-------------------------------
                        4090 * (1B0B) 1048 2765
                        4100 DRAW.TOP.AND.BOTTOM.LINES
1B0B- A2 03             4110         LDX #$03      "HOME" TOKEN
1B0D- 20 E9 1F          4120         JSR DISPLAY.TOKEN.X
1B10- 20 26 D0          4130         JSR X.D026
1B13- 20 29 D0          4140         JSR DISPLAY.FUNCTION.AND.ESCAPE.MAP
1B16- A2 00             4150         LDX #$00
1B18- A4 95             4160         LDY WINDOW.TOP
1B1A- 88                4170         DEY
1B1B- 20 23 18          4180         JSR MOVE.CURSOR.TO.XY
1B1E- A2 4F             4190         LDX #$4F
1B20- A0 3D             4200         LDY #'='      PRINT 79 ='s above window
1B22- 20 BE 20          4210         JSR REPEAT.CHAR.Y.X.TIMES
1B25- A9 2D             4220         LDA #'-'      PRINT 79 -'s below window
1B27- 20 AE 20          4230         JSR DRAW.BOTTOM.LINE
1B2A- 60                4240         RTS
                        4250 *-------------------------------
                        4260 *   Mark that the current file has been changed.
                        4270 *-------------------------------
                        4280 * (1B2B) 104B
                        4290 POST.CHANGE.FLAG
1B2B- AD 6C 0C          4300         LDA FLAG.CURRENT.FILE
1B2E- 09 02             4310         ORA #$02
1B30- 8D 6C 0C          4320         STA FLAG.CURRENT.FILE
1B33- 60                4330         RTS
```

```
                        4340 *-------------------------------
                        4350 * (1B34) 104E 1A09 1C35 2538 2550 2575
                        4360 MULTIPLY.X.BY.Y
1B34- 86 93             4370         STX MUL.X.FACTOR
1B36- 84 94             4380         STY MUL.Y.FACTOR
1B38- A9 00             4390         LDA #$00
1B3A- 85 91             4400         STA M.REG
1B3C- A2 08             4410         LDX #8        8 bits
1B3E- 46 93             4420 .1      LSR MUL.X.FACTOR
1B40- 90 03             4430         BCC .2
1B42- 18                4440         CLC
1B43- 65 94             4450         ADC MUL.Y.FACTOR
1B45- 6A                4460 .2      ROR
1B46- 66 91             4470         ROR M.REG
1B48- CA                4480         DEX
1B49- D0 F3             4490         BNE .1
1B4B- 85 92             4500         STA M.REG+1
1B4D- 60                4510         RTS
                        4520 *-------------------------------
                        4530 * (1B4E) 1051
                        4540 MULTIPLY.PO.BY.P2
1B4E- 20 AE 18          4550         JSR GET.4.PARMS
1B51- B1 9A             4560 .1      LDA (PO),Y
1B53- 99 9E 00          4570         STA P4,Y
1B56- B1 9C             4580         LDA (P2),Y
1B58- 99 91 00          4590         STA M.REG,Y
1B5B- C8                4600         INY
1B5C- C0 01             4610         CPY #$01
1B5E- F0 F1             4620         BEQ .1
1B60- A9 00             4630         LDA #$00
1B62- 85 A0             4640         STA COUNT
1B64- 85 A1             4650         STA Z.A1
1B66- A2 11             4660         LDX #$11
1B68- 18                4670         CLC
1B69- 66 A1             4680 .2      ROR Z.A1
1B6B- 66 A0             4690         ROR COUNT
1B6D- 66 92             4700         ROR M.REG+1
1B6F- 66 91             4710         ROR M.REG
1B71- 90 0D             4720         BCC .3
1B73- A5 A0             4730         LDA COUNT
1B75- 18                4740         CLC
1B76- 65 9E             4750         ADC P4
1B78- 85 A0             4760         STA COUNT
1B7A- A5 A1             4770         LDA Z.A1
1B7C- 65 9F             4780         ADC P5
1B7E- 85 A1             4790         STA Z.A1
1B80- CA                4800 .3      DEX
1B81- D0 E6             4810         BNE .2
1B83- 60                4820         RTS
                        4830 *-------------------------------
                        4840         .ph $1BAC
                        4850 *-------------------------------
                        4860 * (1BAC) 1057 1D1C 2404 263B 2653
                        4870 MOVE.BLOCK.UP
1BAC- A9 06             4880         LDA #$06
1BAE- 20 B4 18          4890         JSR GET.A.PARMS
1BB1- A5 9F             4900         LDA P5
1BB3- F0 0B             4910         BEQ .1
1BB5- 18                4920         CLC
1BB6- 65 9B             4930         ADC P1
1BB8- 85 9B             4940         STA P1
1BBA- A5 9F             4950         LDA P5
1BBC- 65 9D             4960         ADC P3
1BBE- 85 9D             4970         STA P3
1BC0- A4 9E             4980 .1      LDY P4
1BC2- F0 08             4990         BEQ .3
1BC4- 88                5000 .2      DEY
1BC5- B1 9C             5010         LDA (P2),Y
1BC7- 91 9A             5020         STA (PO),Y
1BC9- 98                5030         TYA
1BCA- D0 F8             5040         BNE .2
1BCC- C6 9F             5050 .3      DEC P5
1BCE- 30 0E             5060         BMI .5
1BD0- C6 9B             5070         DEC P1
1BD2- C6 9D             5080         DEC P3
1BD4- 88                5090 .4      DEY
1BD5- B1 9C             5100         LDA (P2),Y
1BD7- 91 9A             5110         STA (PO),Y
1BD9- 98                5120         TYA
1BDA- D0 F8             5130         BNE .4
1BDC- F0 EE             5140         BEQ .3
1BDE- 60                5150 .5      RTS
```

In about the time it takes to read this headline, you can have the Finder up and

Now your favorite program can be ready to go seconds after you flip your Apple IIgs on.

With Applied Engineering's RamKeeper™ card, your IIgs retains stored programs *and* data when you turn your computer off.

RamKeeper powers up to two memory cards simultaneously when your Apple IIgs is off. And battery backup keeps power to the boards even during power failures. Your programs and data are stored in a permanent, "electronic hard disk," always ready to run.

Superior power backup.

Applied Engineering has the most experience in battery-backed memory for Apple computers. We were the first to offer battery-backed memory with our RamFactor™/RamCharger™ combination. Now RamKeeper sets the standard for IIgs memory backup.

RamKeeper lets you keep programs and data in permanent, "electronic hard disk" memory. Turn your Apple IIgs on and you're ready to work.

Our experience shows in the way we designed and built RamKeeper. We used sealed Gel/Cell batteries — far more reliable than Ni-Cads in this application. Ni-Cads lose much of their capacity if they're not discharged periodically. Just when you need them most, Ni-Cads could run out of power.

Our Gel/Cell pack, which is included in our price, gives you up to six hours of total power failure backup. That's about 6 times longer than other systems.

RamKeeper uses a Switching Power Supply — the same technology used by Apple for the IIgs power supply. This design uses energy much more efficiently to keep your Apple running cooler.

Our sealed Gel/Cell battery

stays *outside* your computer case. With other systems, the batteries are installed under the IIGS power supply where a leak could ruin computer circuitry.

Put two memory boards in the same slot.

You might have bought your IIGS with Apple's memory card. But now you want the features of Applied's GS-RAM™ card. RamKeeper efficiently resolves the dilemma.

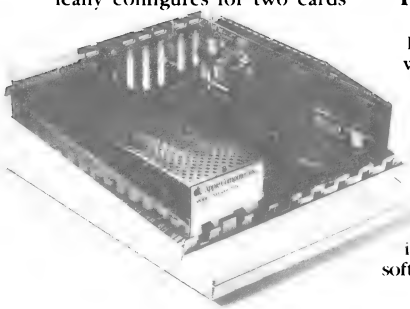
You can use RamKeeper with your current IIGS memory card *and* add another memory card — all in the same slot. Just attach your current memory card to one side of your new RamKeeper card, connect the second card to the other side and plug RamKeeper into the slot.

Of course RamKeeper works fine with just one memory card. But you can use two and still keep Slot 7 open with our optional Slot-Mover.

Makes all of your memory usable memory.

RamKeeper can power up to 16 Meg of memory. Other systems are limited to only 8 Meg. In addition, RamKeeper lets you mix and match different types of cards. For example, you can have a GS-RAM Plus™ using 1 Meg RAM chips and an Apple card using 256K RAM chips. Other systems are limited in the combinations they allow.

RamKeeper firmware automatically configures for two cards when the second card is installed. Other systems make you manually move jumpers.



RamKeeper is easy to install. Just plug it in. Even when you use two memory boards, you don't have jumpers. You can have two memory boards but use only one slot.

It all comes with RamKeeper ... board, Gel/Cell battery pack, easy-to-understand instructions, and Applied's powerful AppleWorks Expander software.

RamKeeper configures memory linearly. Other systems don't, so they create memory gaps that can cause program crashes or keep some programs from using as much as half of your memory.

You easily decide how much memory you'll devote to ROM and to RAM from the IIGS Desk Accessories menu. You can configure Kilobtyes or Megabytes of instant ROM storage for your favorite programs. And you can change ROM size any time without affecting stored files.

Protected from program crashes.

RamKeeper controlling firmware is in an EPROM. A program crash can't take out the operating software. With other systems, operating software is installed in RAM from a floppy. If the program crashes, it can take the operating software with it; and reinstalling the disk-based operating software destroys data in memory.

Verifies data security.

RamKeeper firmware uses optional startup checksums to verify that no data has been lost while power was off. The firmware also runs ROM and RAM memory tests without disturbing data on the card.

Free AppleWorks Enhancement software.

Applied's powerful AppleWorks Enhancement software is free with RamKeeper. It makes AppleWorks faster and far more powerful by eliminating AppleWorks internal memory limits. Word processor limits go from only 7,250 lines to 22,600 lines. Database limits go from 6,350 records to 22,600 records. The clipboard size limit is increased from 255 to 2,042 lines. It even automatically segments large files so you can save them on multiple floppies. No other company expands your IIGS' AppleWorks internal limits.

In addition, the most powerful disk-caching program available comes with the RamKeeper. The cache significantly increases access time to the Apple 3.5 Drive. Most applications will run up to 7 times faster.

The largest maker of Apple expansion boards.

Applied Engineering has sold more expansion boards than anyone else. And we've been in business 8 years, long enough to see the vast majority of our competitors come and go.

All of our products are crafted in the U.S.A. We back RamKeeper with a five year parts and labor warranty. And a 15-day, no questions asked, money back guarantee.

Only $189.00.

See your dealer. Or call 214-241-6060, 9 a.m. to 11 p.m. 7 days. Or send check or money order to Applied Engineering. MasterCard, VISA, C.O.D. welcome. Texas residents add 7% sales tax. Add $10.00 outside U.S.A. Prices subject to change without notice.

APPLIED ENGINEERING™

The Apple enhancement experts.

P.O. Box 5100 • Carrollton, Texas 75011
(214) 241-6060

RamKeeper, RamFactor, GS-RAM, and GS-RAM Plus are trademarks of Applied Engineering

```
                        5160 *--------------------------------
                        5170 * (1BDF) 105A
                        5180 POP.ESCAPE.ROAD.MAP
1BDF- 20 84 1B          5190         JSR MOVE.BLOCK.DOWN
1BE2- E8 0C FD
1BE5- 0C A8 00          5200         .DA ESCAPE.ROAD.MAP,ESCAPE.ROAD.MAP+21,168
1BE8- A9 00             5210         LDA #$00      MARK LAST ENTRY EMPTY
1BEA- 8D 90 0D          5220         STA ESCAPE.ROAD.MAP+168
1BED- 20 29 D0          5230         JSR DISPLAY.FUNCTION.AND.ESCAPE.MAP
1BF0- 60                5240         RTS
                        5250 *--------------------------------
                        5260 * (1BF1) 105D 1862 2389 2728 2ADB
                        5270 WAIT.FOR.SPACE.RETURN.OR.ESCAPE
1BF1- 20 E0 1F          5280         JSR CLEAR.KEYBUF
1BF4- 20 F5 1F          5290         JSR DISPLAY.ON.LINE.23
1BF7- 2E 14             5300         .DA MSG..4   "Press Space Bar to continue"
1BF9- A9 00             5310         LDA #0
1BFB- 85 88             5320         STA FLAG.FOUND.ESCAPE
1BFD- 20 35 1D          5330 .1      JSR AW.KEYIN
1C00- C9 20             5340         CMP #$20
1C02- F0 0F             5350         BEQ .3          ...GOT A SPACE
1C04- C9 0D             5360         CMP #$0D
1C06- F0 0B             5370         BEQ .3          ...GOT <RETURN>
1C08- C9 1B             5380         CMP #$1B
1C0A- F0 05             5390         BEQ .2          ...GOT <ESCAPE>
1C0C- 20 18 18          5400         JSR BEEP.AND.CLEAR.KEYBUF
1C0F- D0 EC             5410         BNE .1          ...ALWAYS
1C11- E6 88             5420 .2      INC FLAG.FOUND.ESCAPE     RETURN FLAG.FOUND.ESCAPE=$01,
1C13- 60                5430 .3      RTS                                          STATUS .NE.
                        5440 *--------------------------------
                        5450 * (1C14) 1CB3 1CDD
1C14-                   5460 PD.CHARCNT .BS 1
                        5470 * (1C15) 1C3F 1CF1
1C15-                   5480 PRNTR.CRLF.FLAG .BS 1
                        5490 * (1C16) 1C6D 2309 2336
1C16-                   5500 I.1C16 .BS 1            'e' if //e, or 'c' if //c
1C17-                   5510 PRNTR.SETUP.STRING .BS 10
                        5520 *--------------------------------
                        5530 * (1C21) 1060 1E22 1E32 1E39 1E48
                        5540 *   Called from KEYIN when Apple-H is pressed,
                        5550 *      and from JMP table at $1060
                        5560 *   (A) = $00        Open Printer
                        5570 *         $FE        Print CRLF
                        5580 *         $FF        Close Printer
                        5590 *        other       A=#chars, P0,P1 is address of text.
                        5600 *--------------------------------
                        5610 PRINTER.DRIVER
1C21- A2 00             5620         LDX #$00
1C23- 86 24             5630         STX MON.CH
1C25- 86 25             5640         STX MON.CV
1C27- AA                5650         TAX
1C28- F0 03             5660         BEQ .1          A=00, Open Printer
1C2A- 4C AD 1C          5670         JMP .6          SOME OTHER FUNCTION
                        5680 *---Open Printer----------------
1C2D- AE 3D 0F          5690 .1      LDX CURRENT.PRINTER.NO
1C30- F0 78             5700         BEQ .5          None, or it would have been 1-3
1C32- CA                5710         DEX             change range to 0,1,2
1C33- A0 24             5720         LDY #36         multiply by 36 bytes per printer
1C35- 20 34 1B          5730         JSR MULTIPLY.X.BY.Y
1C38- A6 91             5740         LDX M.REG
1C3A- BD 73 0F          5750         LDA X.0F73,X
1C3D- 29 20             5760         AND #$20        non-zero means print LF after CR
1C3F- 8D 15 1C          5770         STA PRNTR.CRLF.FLAG
1C42- BD 71 0F          5780         LDA X.0F71,X
1C45- AA                5790         TAX
1C46- BD C6 0F          5800         LDA SLOTROM.MAP,X
1C49- 29 02             5810         AND #$02
1C4B- F0 5D             5820         BEQ .5          Not a printer interface, for sure
1C4D- EC A8 0E          5830         CPX OPEN.PRINTER.FLAG
1C50- F0 58             5840         BEQ .5
1C52- 8E A8 0E          5850         STX OPEN.PRINTER.FLAG
1C55- 8A                5860         TXA
1C56- 09 C0             5870         ORA #$C0
1C58- 85 37             5880         STA CSWH
1C5A- A9 00             5890         LDA #$00
1C5C- 85 36             5900         STA CSWL
1C5E- AD 0A 1D          5910         LDA HANDLE.PRNTR.SETUP.STRING
1C61- 85 9A             5920         STA P0
1C63- AD 0B 1D          5930         LDA HANDLE.PRNTR.SETUP.STRING+1
1C66- 85 9B             5940         STA P1
```

```
1C68- A6 91     5950         LDX M.REG
1C6A- BD 70 0F  5960         LDA X.0F70,X
1C6D- CD 16 1C  5970         CMP I.1C16
1C70- D0 0F     5980         BNE .2
1C72- AD 0C 1D  5990         LDA HANDLE.0F61
1C75- 18        6000         CLC
1C76- 65 91     6010         ADC M.REG
1C78- 85 9A     6020         STA P0
1C7A- AD 0D 1D  6030         LDA HANDLE.0F61+1
1C7D- 69 00     6040         ADC #$00
1C7F- 85 9B     6050         STA P1
1C81- A0 00     6060 .2      LDY #$00
1C83- B1 9A     6070         LDA (P0),Y
1C85- F0 23     6080         BEQ .5
1C87- 85 9C     6090         STA P2
1C89- 8D 82 C0  6100         STA ROM.D000
1C8C- C8        6110 .3      INY
1C8D- B1 9A     6120         LDA (P0),Y
1C8F- C9 09     6130         CMP #$09
1C91- D0 10     6140         BNE .4
1C93- A9 89     6150         LDA #$89
1C95- AE EF 0F  6160         LDX FLAG.APPLE.2C
1C98- F0 09     6170         BEQ .4          ...not an Apple //c
1C9A- AE A8 0E  6180         LDX OPEN.PRINTER.FLAG
1C9D- E0 02     6190         CPX #$02
1C9F- D0 02     6200         BNE .4
1CA1- A9 01     6210         LDA #$01
1CA3- 20 ED FD  6220 .4      JSR MON.COUT
1CA6- C4 9C     6230         CPY P2
1CA8- 90 E2     6240         BCC .3
1CAA- 4C 03 1D  6250 .5      JMP .13
                6260 *-------------------------------
1CAD- C9 FE     6270 .6      CMP #$FE
1CAF- F0 33     6280         BEQ .11         A=FE, Print CRLF
1CB1- B0 4B     6290         BCS .12         A=FF, Close Printer
1CB3- 8D 14 1C  6300         STA PD.CHARCNT     A=number chars to print
1CB6- 20 B2 18  6310         JSR GET.2.PARMS
1CB9- AD A8 0E  6320         LDA OPEN.PRINTER.FLAG
1CBC- F0 45     6330         BEQ .13
1CBE- 8D 82 C0  6340         STA ROM.D000
1CC1- A0 00     6350         LDY #$00
1CC3- B1 9A     6360 .7      LDA (P0),Y
1CC5- C9 81     6370         CMP #$81        Change "Apple" to "@"
1CC7- D0 04     6380         BNE .8          ...not an apple
1CC9- A9 40     6390         LDA #$40
1CCB- D0 06     6400         BNE .9          ...always
1CCD- C9 8D     6410 .8      CMP #$8D        change <RETURN> to <SPACE>
1CCF- D0 02     6420         BNE .9          ...not <RETURN>
1CD1- A9 20     6430         LDA #$20
1CD3- C9 09     6440 .9      CMP #$09        change 09 (ctrl-I) to 89
1CD5- D0 02     6450         BNE .10         ...not 09
1CD7- A9 89     6460         LDA #$89
1CD9- 20 ED FD  6470 .10     JSR MON.COUT       out to the printer, finally
1CDC- C8        6480         INY
1CDD- CC 14 1C  6490         CPY PD.CHARCNT
1CE0- 90 E1     6500         BCC .7          ...more to print
1CE2- B0 1F     6510         BCS .13         ...finished
                6520 *---Print CRLF-------------------
1CE4- AD A8 0E  6530 .11     LDA OPEN.PRINTER.FLAG
1CE7- F0 1A     6540         BEQ .13
1CE9- 8D 82 C0  6550         STA ROM.D000
1CEC- A9 0D     6560         LDA #$0D
1CEE- 20 ED FD  6570         JSR MON.COUT
1CF1- AD 15 1C  6580         LDA PRNTR.CRLF.FLAG
1CF4- F0 0D     6590         BEQ .13
1CF6- A9 0A     6600         LDA #$0A
1CF8- 20 ED FD  6610         JSR MON.COUT
1CFB- 4C 03 1D  6620         JMP .13
                6630 *---Close Printer----------------
1CFE- A9 00     6640 .12     LDA #$00
1D00- 8D A8 0E  6650         STA OPEN.PRINTER.FLAG
1D03- AD 83 C0  6660 .13     LDA RAM.D000
1D06- AD 83 C0  6670         LDA RAM.D000
1D09- 60        6680         RTS
                6690 *-------------------------------
                6700 * (1D0A,B) 1C5E 1C63
1D0A- 17 1C     6710 HANDLE.PRNTR.SETUP.STRING .DA PRNTR.SETUP.STRING
                6720 * (1D0C,D) 1C72 1C7A
1D0C- 61 0F     6730 HANDLE.0F61 .DA X.0F61
                6740 *-------------------------------
```

```
                    6750 * (1D0E) 1E10
1D0E-               6760         .BS 1
                    6770 *--------------------------------
                    6780 * (1D0F) 1063
                    6790 PUSH.ESCAPE.ROAD.MAP
1D0F- 20 B2 18      6800         JSR GET.2.PARMS
1D12- A5 9A         6810         LDA P0
1D14- 8D 2A 1D      6820         STA .1         FILL IN ADDRESS
1D17- A5 9B         6830         LDA P1
1D19- 8D 2B 1D      6840         STA .1+1
1D1C- 20 AC 1B      6850         JSR MOVE.BLOCK.UP
1D1F- FD 0C E8
1D22- 0C A8 00      6860         .DA ESCAPE.ROAD.MAP+21,ESCAPE.ROAD.MAP,168
1D25- 20 F8 1E      6870         JSR MOVE.STRING
1D28- E8 0C         6880         .DA ESCAPE.ROAD.MAP
1D2A- 00 00         6890 .1      .DA 0000     FILLED IN
1D2C- 20 29 D0      6900         JSR DISPLAY.FUNCTION.AND.ESCAPE.MAP
1D2F- 60            6910         RTS
                    6920 *--------------------------------
                    6930         .ph $1E80
                    6940 *--------------------------------
                    6950 * (1E80) 1069 1911 193C 19B2 2025 2612 261A
                    6960 MOVE.CURSOR.TO.TCOL.TROW
1E80- AE AD 0E      6970         LDX TCOL
1E83- AC AE 0E      6980         LDY TROW
1E86- 20 23 18      6990         JSR MOVE.CURSOR.TO.XY
1E89- 60            7000         RTS
                    7010 *--------------------------------
                    7020 * (1E8A) 106C
                    7030 SAVE.GOTO.XY
1E8A- 8E AD 0E      7040         STX TCOL
1E8D- 8C AE 0E      7050         STY TROW
1E90- 20 23 18      7060         JSR MOVE.CURSOR.TO.XY
1E93- 60            7070         RTS
                    7080 *--------------------------------
                    7090         .ph $1EB4
                    7100 *--------------------------------
                    7110 * (1EB4) 106F
                    7120 MAP.LOWER.TO.UPPER
1EB4- C9 61         7130         CMP #'a'
1EB6- 90 06         7140         BCC .1         ...not lower-case
1EB8- C9 7B         7150         CMP #'z'+1
1EBA- B0 02         7160         BCS .1         ...not lower-case
1EBC- 29 DF         7170         AND #$DF       flip to upper-case
1EBE- 60            7180 .1      RTS
                    7190 *--------------------------------
                    7200         .ph $1F3E
                    7210 *--------------------------------
                    7220 *   Display string (P2,P3) at (P0,P1)
                    7230 *      JSR DISPLAY.AT
                    7240 *      .DA #column,#line
                    7250 *      .DA string.address
                    7260 *
                    7270 *   If column value is negative but not $FF, then add $94
                    7280 *     which clears bit 7 and adds 20
                    7290 *   If column value is $FF then center the string in 80 columns
                    7300 *--------------------------------
                    7310 * (1F3E) 107B 183A 185B 200C 25E5 2713 271A 2ACD 2B70 2B77
                    7320 * (1F3E) 2B8C 2B93 2BA1 2BA8 2D37 2D3E
                    7330 DISPLAY.AT
1F3E- 20 AE 18      7340         JSR GET.4.PARMS
1F41- A9 05         7350         LDA #$05       Build string to set cursor position
1F43- 8D 00 0A      7360         STA STR.A
1F46- A5 9A         7370         LDA P0
1F48- 8D 01 0A      7380         STA STR.A+1
1F4B- A5 9B         7390         LDA P1
1F4D- 8D 02 0A      7400         STA STR.A+2
1F50- A5 9C         7410         LDA P2
1F52- 18            7420         CLC
1F53- 69 01         7430         ADC #1         Build address of 1st char after length
1F55- 8D B2 0E      7440         STA X.0EB2
1F58- A5 9D         7450         LDA P3
1F5A- 69 00         7460         ADC #0
1F5C- 8D B3 0E      7470         STA X.0EB3
1F5F- A0 00         7480         LDY #0         Get length of string
1F61- B1 9C         7490         LDA (P2),Y
1F63- 20 A0 1F      7500         JSR TRUNCATE.TO.79.IF.OVER.80
1F66- 8D B4 0E      7510         STA X.0EB4
1F69- AD 01 0A      7520         LDA STR.A+1    Was P0 positive, meaning absolute column?
1F6C- 10 17         7530         BPL .2         ...yes
```

```
1F6E- C9 FF    7540         CMP #$FF
1F70- D0 0D    7550         BNE .1        ...not centering, so add $94
1F72- A9 50    7560         LDA #80       $FF means to center in 80 columns
1F74- 38       7570         SEC           (80-length)/2
1F75- ED B4 0E 7580         SBC X.0EB4
1F78- 4A       7590         LSR
1F79- 8D 01 0A 7600         STA STR.A+1
1F7C- 4C 85 1F 7610         JMP .2
               7620 *-------------------------------
1F7F- 18       7630 .1      CLC
1F80- 69 94    7640         ADC #$94
1F82- 8D 01 0A 7650         STA STR.A+1
1F85- 20 A9 1E 7660 .2      JSR POINT.PSTR.AT.0A00
1F88- A9 03    7670         LDA #3
1F8A- 20 D1 14 7680         JSR DISPLAY.STRING
1F8D- AD B2 0E 7690         LDA X.0EB2
1F90- 85 80    7700         STA PSTR
1F92- AD B3 0E 7710         LDA X.0EB3
1F95- 85 81    7720         STA PSTR+1
1F97- AD B4 0E 7730         LDA X.0EB4
1F9A- F0 03    7740         BEQ .3        null string
1F9C- 20 D1 14 7750         JSR DISPLAY.STRING
1F9F- 60       7760 .3      RTS
               7770 *-------------------------------
               7780 * (1FA0) 1F63 20A7 20C2
               7790 TRUNCATE.TO.79.IF.OVER.80
1FA0- C9 51    7800         CMP #81
1FA2- 90 02    7810         BCC .1
1FA4- A9 4F    7820         LDA #79
1FA6- 60       7830 .1      RTS
               7840 *-------------------------------
               7850         .ph $1FE9
               7860 *-------------------------------
               7870 * (1FE9) 108A 181A 1970 19B7 19C2 1A1E 1A97 1AAB 1AC3 1AD8
               7880 * (1FE9) 1B0D 2015 2B5F
               7890 DISPLAY.TOKEN.X
1FE9- 8E 00 0A 7900         STX STR.A
1FEC- 20 A9 1E 7910         JSR POINT.PSTR.AT.0A00
1FEF- A9 01    7920         LDA #1
1FF1- 20 D1 14 7930         JSR DISPLAY.STRING
1FF4- 60       7940         RTS
               7950 *-------------------------------
               7960 *   JSR DISPLAY.ON.LINE.23
               7970 *   .DA string
               7980 *
               7990 *   1.  Display string starting at col. 0, line 23
               8000 *   2.  Clear rest of the line
               8010 *   3.  If Z.89 is $00, display "xxxxK Avail.";
               8020 *       otherwise, display "Apple-? for Help"
               8030 *   4.  Move cursor to end of the string.
               8040 *-------------------------------
               8050 * (1FF5) 1093 118D 186C 1931 1A45 1BF4 20D6 25F3
               8060 DISPLAY.ON.LINE.23
1FF5- 20 B2 18 8070         JSR GET.2.PARMS
1FF8- A5 9A    8080         LDA P0
1FFA- 8D 11 20 8090         STA .1        store address in parameter below
1FFD- A5 9B    8100         LDA P1
1FFF- 8D 12 20 8110         STA .1+1
2002- B1 9A    8120         LDA (P0),Y    length of string
2004- 8D AD 0E 8130         STA TCOL      later position cursor to end of string
2007- A9 17    8140         LDA #23          on line 23
2009- 8D AE 0E 8150         STA TROW
200C- 20 3E 1F 8160         JSR DISPLAY.AT
200F- 00 17    8170         .DA #0,#23    Column 0, line 23
2011- 00 00    8180 .1      .DA 0000      String address, filled in
2013- A2 01    8190         LDX #$01      "Clear to End of Line" token
2015- 20 E9 1F 8200         JSR DISPLAY.TOKEN.X   Print the token
2018- A5 89    8210         LDA Z.89
201A- F0 06    8220         BEQ .2        ...display "xxxxK Avail."
201C- 20 37 18 8230         JSR SHOW.HELP.STRING   Display "Apple-? for Help"
201F- 4C 25 20 8240         JMP .3
2022- 20 44 D0 8250 .2      JSR DISPLAY.K.AVAIL   Display "xxxxK Avail."
2025- 20 80 1E 8260 .3      JSR MOVE.CURSOR.TO.TCOL.TROW
2028- 60       8270         RTS
               8280 *-------------------------------
```

Klaxon Sound Effect.....Robert C. Moore

Here is an interesting little sound effect generator, which you might like to use to call attention to an operator error. It uses the simple Apple speaker, so it is compatible with all of the many models in the Apple II series and even the clones.

Lines 1110-1130 call on the SOUNDS subroutine to generate two bursts of sound. The first burst combines plays a higher note, the second a lower note. Both notes are combined inside SOUNDS with a pitch that is in between the two. The total effect sounds a little like a klaxon to me.

Bob S-C inserted line 1100, which turns the dee-daw sound into dee-daw-dee-daw. You might also enjoy experimenting with different pitches and durations. You can change the intermediate pitch by change the value 63 in lines 1190 and 1240.

You might notice that DURATION2 never gets initialized to anything. If you want to, you could store 0 there by inserting these lines:

```
1191     LDA #0
1192     STA DURATION2
```

However, you will not be able to tell the difference in how it sounds. The total time the horn blows is DURATION1 times 256, less up to 255 if DURATION2 starts out non-zero. In the worst case, with DURATION2 starting at 1, the horn will finish blowing about 5% sooner.

```
                    1010 *--------------------------------
                    1020 *   from Robert C. Moore, Laurel, Maryland.
                    1030 *--------------------------------
00-                 1040 DURATION1  .EQ 0
01-                 1050 DURATION2  .EQ 1
02-                 1060 PITCH      .EQ 2
                    1070 *--------------------------------
C030-               1080 SPEAKER    .EQ $C030
                    1090 *--------------------------------
0800- 20 03 08      1100 BOBSC  JSR KLAXON
0803- A0 32         1110 KLAXON LDY #50       FIRST PRIMARY PITCH
0805- 20 0A 08      1120        JSR SOUNDS
0808- A0 50         1130        LDY #80       SECOND PRIMARY PITCH
                    1140 *--------------------------------
                    1150 SOUNDS
080A- 84 02         1160        STY PITCH     SAVE CALLER'S Y-PITCH
080C- A9 14         1170        LDA #20       LENGTH OF SOUND
080E- 85 00         1180        STA DURATION1
0810- A2 3F         1190        LDX #63       SECONDARY PITCH
                    1200 *--------------------------------
0812- CA            1210 .2     DEX           COUNT DOWN SECONDARY CYCLE
0813- D0 05         1220        BNE .3        ...NOT TIME FOR CLICK YET
0815- 2C 30 C0      1230        BIT SPEAKER   ...CLICK NOW
0818- A2 3F         1240        LDX #63       START ANOTHER CYCLE
                    1250 *--------------------------------
081A- 88            1260 .3     DEY           COUNT DOWN PRIMARY CYCLE
081B- D0 05         1270        BNE .4        ...NOT TIME FOR CLICK YET
081D- 2C 30 C0      1280        BIT SPEAKER   ...CLICK NOW
0820- A4 02         1290        LDY PITCH     START ANOTHER CYCLE
                    1300 *--------------------------------
0822- C6 01         1310 .4     DEC DURATION2     256*20 TIMES ALTOGETHER
0824- D0 EC         1320        BNE .2
0826- C6 00         1330        DEC DURATION1
0828- D0 E8         1340        BNE .2
082A- 60            1350        RTS
```

AppleWorks Segment Functions.....Steve Stephenson

I have been disassembling and patching AppleWorks since 1984 for CheckMate Technology, and have learned a few things about it. One interesting task was trying to figure out what each one of the segments in the SEG.M0 and SEG.M1 files does. As near as I can tell, here is the breakdown for AppleWorks 1.3:

```
Seg.01 DB Common                Seg.24 SS Common
Seg.02 DB Main                  Seg.25 SS Main
Seg.03 DB Help                  Seg.26 SS Help
Seg.04 DB SingleLayout          Seg.27 SS Print
Seg.05 DB MultipleLayout        Seg.28 SS Layout
Seg.06 DB ChangeNames           Seg.29 SS Edit
Seg.07 DB Search                Seg.30 SS ToClipboard
Seg.08 DB Sort                  Seg.31 SS Options
Seg.09 DB Report Main
Seg.10 DB Report Util           Seg.32 DeskTop Manager
Seg.11 DB Report New            Seg.33 File Loader
Seg.12 DB Report Layout         Seg.34 Printer Setups
Seg.13 DB Report Open           Seg.35 Import DIF for DB
Seg.14 DB Report Output         Seg.36 File Saver
Seg.15 -doesn't exist-          Seg.37 Import Quick Files for DB
                                Seg.38 Import ASCII Text for WP
Seg.16 WP Common                Seg.39 Import VisiCalc for SS
Seg.17 WP Main
Seg.18 WP Help                  Seg.40 Main Help
Seg.19 WP Print
Seg.20 -doesn't exist-          Seg.41 Import Text for DB
Seg.21 -doesn't exist-          Seg.42 Import DIF for SS
Seg.22 SANE
Seg.23 SANE                     Seg.43 Disk Formatter
```

I also wrote a program which will break out the segments from the SEG.M0 and SEG.M1 files into individual binary files, each with their proper SEG number, load address, and length.

There is no fancy prefix handling in the program, so the two source files and all the resultant files must all land in the same directory. I use a RAM disk of at least 600 blocks, for speed. There is also not any fancy error handling. If any error occurs, the hexadecimal error code returned by MLI will be displayed. If not, 00 will be displayed when it's finished.

The monthly AAL disk also includes a slightly different version of the Segment Splitter, which displays a line for each segment including the segment number, the file offset, starting address, and length.

```
                1020 *--------------------------------
                1030 *  AppleWorks Segment Splitter
                1040 *       by Steve Stephenson
                1050 *--------------------------------
BF00-           1060 MLI        .EQ $BF00
BF0F-           1070 ERRCODE    .EQ $BF0F
C0-             1080 CREATE     .EQ $C0       ProDOS MLI Functions
C8-             1090 OPEN       .EQ $C8
CA-             1100 READ       .EQ $CA
CB-             1110 WRITE      .EQ $CB
CC-             1120 CLOSE      .EQ $CC
CE-             1130 SETMARK    .EQ $CE
                1140 *--------------------------------
```

```
                    1150         .MA MLI       Function, Parameter List
                    1160         JSR MLI
                    1170         .DA #]1,]2
                    1180         BCC :1        No Error
                    1190         JMP ERROR
                    1200 :1
                    1210         .EM
                    1220 *--------------------------------
                    1230         .MA STR
                    1240         .DA #:1-*-1   Number of bytes in string
                    1250         .AS -*]1*     String itself
                    1260 :1
                    1270         .EM
                    1280 *--------------------------------
0300-               1290 BUFFER.INDEX .EQ $300
06-                 1300 FILE.CNT .EQ $6
                    1310 *--------------------------------
                    1320         .OR $1000
1000-               1330         .TF B.SEG.SPLITTER
                    1340 *--------------------------------
                    1350 SEGMENT.SPLITTER
1000- A9 01         1360         LDA #1
1002- 85 06         1370         STA FILE.CNT
                    1380 PRELOOP
1004-               1390         >MLI OPEN,P.OPEN.SEGMX    open seg.m0/1
100F- AD 38 11      1400         LDA P.OPEN.SEGMX+5        file refnum
1012- 8D 3A 11      1410         STA P.READ.INDEX+1
1015- 8D 47 11      1420         STA P.READ.SEGNN+1
1018- 8D 42 11      1430         STA P.SET.MARK+1
101B-               1440         >MLI READ,P.READ.INDEX    read the index
                    1450 *--------------------------------
                    1460 MAINLOOP
1026- A5 06         1470         LDA FILE.CNT find offset to this seg
1028- 0A            1480         ASL          by multiplying by three
1029- 65 06         1490         ADC FILE.CNT
102B- AA            1500         TAX
102C- BD 00 03      1510         LDA BUFFER.INDEX,X   get starting point in file
102F- 8D 43 11      1520         STA P.SET.MARK+2     for SETMARK call
1032- BD 01 03      1530         LDA BUFFER.INDEX+1,X
1035- 8D 44 11      1540         STA P.SET.MARK+3
1038- BD 02 03      1550         LDA BUFFER.INDEX+2,X
103B- 8D 45 11      1560         STA P.SET.MARK+4
103E- 0D 44 11      1570         ORA P.SET.MARK+3     if 000000, no segment exists
1041- 0D 43 11      1580         ORA P.SET.MARK+2
1044- D0 03         1590         BNE .1               ...there is a segment
1046- 4C DF 10      1600         JMP NEXT             ...not one, go to next one
1049- BD 03 03      1610 .1      LDA BUFFER.INDEX+3,X
104C- 1D 04 03      1620         ORA BUFFER.INDEX+4,X
104F- 1D 05 03      1630         ORA BUFFER.INDEX+5,X
1052- D0 03         1640         BNE .2
1054- 4C DF 10      1650         JMP NEXT
                    1660 *---Compute length of segment----
1057- 38            1670 .2      SEC
1058- BD 03 03      1680         LDA BUFFER.INDEX+3,X start of next seg
105B- FD 00 03      1690         SBC BUFFER.INDEX,X   - start of this
105E- 8D 4A 11      1700         STA P.READ.SEGNN+4   = length of this
1061- 8D 64 11      1710         STA P.WRIT.SEGNN+4
1064- BD 04 03      1720         LDA BUFFER.INDEX+4,X
1067- FD 01 03      1730         SBC BUFFER.INDEX+1,X
106A- 8D 4B 11      1740         STA P.READ.SEGNN+5
106D- 8D 65 11      1750         STA P.WRIT.SEGNN+5
1070- 0D 4A 11      1760         ORA P.READ.SEGNN+4   if length = 0000, go to next
1073- D0 03         1770         BNE .3
1075- 4C DF 10      1780         JMP NEXT
                    1790 *---Read the Segment-------------
1078-               1800 .3      >MLI SETMARK,P.SET.MARK  move to the segment
1083-               1810         >MLI READ,P.READ.SEGNN   read it into main BUFFER
                    1820 *---Compute Start Address--------
108E- 38            1830         SEC
108F- AD 00 1A      1840         LDA BUFFER           ending addr of this seg
1092- ED 4A 11      1850         SBC P.READ.SEGNN+4 - length of this segment
1095- AA            1860         TAX                  = equals starting addr
1096- AD 01 1A      1870         LDA BUFFER+1
1099- ED 4B 11      1880         SBC P.READ.SEGNN+5
109C- 8D 54 11      1890         STA P.CREATE+6
109F- 8A            1900         TXA                  round to an even page
10A0- F0 05         1910         BEQ .4
10A2- EE 54 11      1920         INC P.CREATE+6
10A5- A9 00         1930         LDA #0
10A7- 8D 53 11      1940 .4      STA P.CREATE+5
```

```
                        1950 *---Create the Segment File------
10AA-                   1960         >MLI CREATE,P.CREATE  create an individual seg.xx
10B5-                   1970         >MLI OPEN,P.OPEN.SEGNN    open a seg.xx
10C0- AD 5F 11          1980         LDA P.OPEN.SEGNN+5        refnum of file
10C3- 8D 61 11          1990         STA P.WRIT.SEGNN+1
10C6- 8D 69 11          2000         STA P.CLOSE+1
10C9-                   2010         >MLI WRITE,P.WRIT.SEGNN   write out a seg.xx
10D4-                   2020         >MLI CLOSE,P.CLOSE        close seg.xx
                        2030 *-------------------------------
                        2040 NEXT
10DF- EE 32 11          2050         INC SNUMLO
10E2- AD 32 11          2060         LDA SNUMLO
10E5- C9 BA             2070         CMP #"9"+1
10E7- 90 08             2080         BCC .1
10E9- EE 31 11          2090         INC SNUMHI
10EC- A9 B0             2100         LDA #"0"
10EE- 8D 32 11          2110         STA SNUMLO
10F1- E6 06             2120 .1      INC FILE.CNT
10F3- A5 06             2130         LDA FILE.CNT
10F5- C9 0A             2140         CMP #10
10F7- 90 06             2150         BCC .2
10F9- F0 07             2160         BEQ .3
10FB- C9 2C             2170         CMP #44
10FD- B0 0C             2180         BCS .4
10FF- 4C 26 10          2190 .2      JMP MAINLOOP
1102- EE 2B 11          2200 .3      INC NAME.M1
1105- 20 14 11          2210         JSR CLOSEALL
1108- 4C 04 10          2220         JMP PRELOOP
110B- 20 14 11          2230 .4      JSR CLOSEALL
                        2240 *-------------------------------
                        2250 ERROR
110E- AD 0F BF          2260         LDA ERRCODE
1111- 20 DA FD          2270         JSR $FDDA
                        2280 *-------------------------------
                        2290 CLOSEALL
1114- A9 00             2300         LDA #0
1116- 8D 69 11          2310         STA P.CLOSE+1
1119-                   2320         >MLI CLOSE,P.CLOSE
1124- 60                2330         RTS
                        2340 *-------------------------------
1125-                   2350 NAME.MX  >STR "SEG.M0"
112B-                   2360 NAME.M1   .EQ *-1
                        2370 *-------------------------------
112C-                   2380 NAME.NN  >STR "SEG.01"
1132-                   2390 SNUMLO    .EQ *-1
1131-                   2400 SNUMHI    .EQ *-2
                        2410 *-------------------------------
                        2420 *   Parameter Lists for MLI Calls
                        2430 *-------------------------------
1133- 03 25 11
1136- 00 12 00 2440 P.OPEN.SEGMX .DA #3,NAME.MX,BUFFER.SEGMX,#0
1139- 04 00 00
113C- 03 8C 00
113F- 00 00    2450 P.READ.INDEX .DA #4,#00,BUFFER.INDEX,140,0000
1141- 02 00 00
1144- 00 00    2460 P.SET.MARK   .DA #2,#00,<000000
1146- 04 00 00
1149- 1A 00 00
114C- 00 00    2470 P.READ.SEGNN .DA #4,#00,BUFFER,0000,0000
114E- 07 2C 11
1151- C3 06 00
1154- 00 00 00
1157- 00 00 00 2480 P.CREATE     .DA #7,NAME.NN,#$C3,#$06,0000,#00,0000,0000
115A- 03 2C 11
115D- 00 16 00 2490 P.OPEN.SEGNN .DA #3,NAME.NN,BUFFER.SEGNN,#00
1160- 04 00 00
1163- 1A 00 00
1166- 00 00    2500 P.WRIT.SEGNN .DA #4,#00,BUFFER,0000,0000
1168- 01 00    2510 P.CLOSE      .DA #1,#00
                        2520 *-------------------------------
                        2530              .DUMMY
116A-                   2540              .BS *+255/256*256-*   force to page boundary
1200-                   2550 BUFFER.SEGMX .BS $400
1600-                   2560 BUFFER.SEGNN .BS $400
1A00-                   2570 BUFFER       .EQ *
                        2580              .ED
                        2590 *-------------------------------
```

More on AuxTypes in ProDOS CATALOGs.....Bob Sander-Cederlof

Last month I presented a patch for BASIC.SYSTEM and another for SCASM.SYSTEM to cause the AuxType to be displayed for all file types, rather than just for TXT and BIN files. My patch was not quite sufficient, it turns out. My patch only worked for files whose type is a named type, displayed in the catalog with a three-letter name. Files with types displayed as a hexadecimal number still do not show the AuxType. Since I "tested" my patches in directories which only had named filetypes, I did not see the error.

That was the bad news; the good news is that this month I will complete the job, and give you another simple patch to cause the rest of the AuxTypes to be displayed.

In BASIC.SYSTEM you need to patch $A4F6 (which was $44) to $0E. You can do this with a POKE 42230,14. To put it back, POKE 42230,68. Or to make both last month's and this new change, permanently, do this:

```
BLOAD BASIC.SYSTEM,TSYS,A$2000
POKE 12051,17
POKE 12022,14
BSAVE BASIC.SYSTEM,TSYS,A$2000
```

The corresponding change in SCASM.SYSTEM is a little more difficult, because of the various versions out there. If you found last month's location, this new one is a JMP instruction about 14 lines before that one. Disassembling around $AFD0, you should see:

```
        CA        DEX
        CA        DEX
        CA        DEX
        CA        DEX
        10 F5     BPL ...
        20 XX XX  JSR ...
        4C XX XX  JMP ...          patch the JMP address to point
        CA        DEX                 to the BIT below.
        BD XX XX  LDA ...,X
        20 XX XX  JSR ...
        D0 F7     BNE ...
here    2C XX XX  BIT ...
```

You need to determine the address of the BIT instruction, and make the JMP jump to it. If you are with me this far, I am sure you can figure out the details.

New Version 1.2 of BASIC.SYSTEM.....Bob Sander-Cederlof

When I receive a new version of something from Apple, my first impulse is to try to find out exactly what they changed. Especially, when for the first time in four years they update a program so important as BASIC.SYSTEM. And especially when there have been excellent articles published in the last four years clearly describing definite bugs, patches, and work-arounds.

I was very disappointed this morning after carefully analyzing the new version 1.2 of BASIC.SYSTEM. I started by BLOADing the old version 1.1 and then copying it into bank 2 of my IIgs. Then I BLOADed the new version 1.2 and used the monitor V-command to compare the two. There were a total of 24 bytes changed. Thirteen were inside the parameter block for a Get-File-Info call, so their value is irrelevant. One is a byte that is never referenced in any way. Three bytes were changed in the title screen, so that you see "1.2" instead of "1.1", and "COPYRIGHT APPLE 1983-87" instead of "COPYRIGHT APPLE, 1983-84". That leaves only seven bytes in the total update whose change has any significance. They have not fixed even ONE of the many published problems in BASIC.SYSTEM!

So what did they fix? The description sheet that came with the update said they were trying to fix a bug in the CATALOG command. A variable they call TOTENT, which happens to be at BCB9-BCBA, is used for a counter to control the loop which displays files names and info. When the directory is first opened the total number of files in the directory is copied into TOTENT. The original intention of the programmer was to decrement TOTENT after reading each file entry in the directory. When the counter reaches "zero" the catalog should be finished. Unfortunately, the program did not decrement the counter properly.

To make matters worse, the new code in version 1.2 does not fix the original bug. Instead, the patch just omits testing TOTENT altogether. Now if you have a long directory, delete most of the files leaving just a few file names in the first few entries, and CATALOG it in BASIC.SYSTEM, it will read all of the entries anyway. No real problem, just spins the disk a fraction of a second longer.

The original bug was not a very serious problem either. It only failed when the total number of active files in a directory was a multiple of 256, which seldom happens. In fact, it seldom happens that there are that many files in any one directory, because so many of the utilities and even AppleWorks get confused with large directories. The symptom you would see if you had exactly 256 files in a directory, as I understand it, is that you would get an "OUT OF DATA" error message at the end of the catalog instead of the "number of blocks" line. I suppose that could be unnerving, so the bug should be removed if possible.

The faulty decrement code is at the end of the Read Next Catalog Entry subroutine, at $B215, and looks like this:

```
B215- DEC $BCB9
B218- BNE $B21D
B21A- DEC $BCBA
B21D- RTS
```

If the initial number in BCB9 (low byte) and BCBA (high byte) is not a multiple of 256, this code will always result in $BCBA going negative when the total value has been counted down. But if the initial value IS a multiple of 256, it will take an extra 256 times to count it down to a negative value in $BCBA. The end-of-loop test code is at $B09E:

```
B09E- LDA $BCBA
B0A1- BPL $B078
```

The correct way to decrement the 16-bit value is like this:

```
    LDA $BCB9
    BNE .1
    DEC $BCBA
.1  DEC $BCB9
```

This code results in both bytes being zero when it is counted all the way down. Code to test the TOTENT variable for zero already exists at the top of the loop in BASIC.SYSTEM:

```
B070- LDA $BCB9
B073- ORA $BCBA
B076- BEQ $B0A3
```

A little re-structuring of the code would result in even fewer bytes being required to do the decrement and loop control correctly. Instead, we have this strange wipe-out instead. Apple went further, and changed the branch at $B076 to two NOP's, and the error branch following the call to Read Next Catalog Entry to terminate the catalog without error. Very interesting. I wonder if they know something I don't? Maybe the value in the directory which we get TOTENT from is sometimes incorrect? Maybe it is sometimes 0000 when there are really files? Why else NOP-out the instruction at $B076? Well, I have never yet noticed such a problem. Have you? Notice that, with these patches, if you get a disk error when reading a directory block CATALOG will terminate without reporting the error: you just will not see the rest of the files.

The description also claimed to fix a problem which caused the CATALOG to prematurely terminate if a <space> was pressed after a control-S. I have never noticed any such problem in the old version, and was unable to make it happen today. But sure enough, it doesn't fail that way in the new version either. After all, they didn't change any of that code anyway!

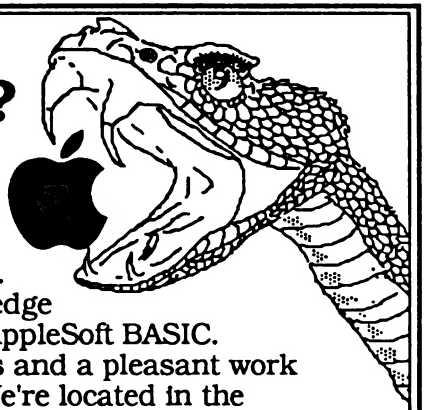
Why didn't Apple confer with Ken Kashmarek, Cecil Fretwell, Sandy Mossberg, Don Worth, Pieter Lechner, Dennis Doms or others who have been so carefully analyzing ProDOS and BASIC.SYSTEM over the last four years?

Anyway, after all the above is said, maybe you still wish you had version 1.2. If so, you can turn version 1.1 into 1.2 like this:

```
]BLOAD BASIC.SYSTEM,TSYS,A$2000
]CALL-151
*2282:B2            (was B1          )
*229A:A0            (was AC          )
*22A2:B7            (was B4          )
*3A76:EA EA         (was F0 28       )
*3A7C:26            (was 3A          )
*3A9E:EA A9 FF D0   (was AD BA BC 10)
*3D0G
]BSAVE BASIC.SYSTEM,TSYS,A$2000
```

The new version I have described above came on the newest IIgs system disk. This also included an update to ProDOS-8 called version 1.6. I personally never saw version 1.5, so I compared 1.6 to 1.4. There were way too many differences to determine what they really changed. Most of the differences appear to be due to a reassembly, so that the addresses of nearly everything internal have changed. A list of changes from 1.5 to 1.6 was included with the other documentation, but as I said I never saw version 1.5. Furthermore, after my experience with verson 1.2 of BASIC.SYSTEM, I don't know what to believe. The one major fix they claim in 1.6 is only significant in the IIgs, and has to do with Desk Accessories determining whether ProDOS MLI was busy or not when the Desk Accessory was called up. If you are not using a IIgs, I think I would stick with version 1.4.

Do You Have Apple Knowledge?



If you do, Applied Engineering would like to put your knowledge to work. We're looking for someone to fill a position in our Technical Support group. You must have a strong working knowledge of AppleWorks, ProDOS, DOS 3.3, and AppleSoft BASIC. Applied Engineering offers good benefits and a pleasant work environment. Office is non-smoking. We're located in the Carrollton area.

So if you've got the knowledge and want to sink your teeth into a position with an ever-expanding company, give us a call at **(214) 241-6060.**

The Apple IIx Wish-O-Gram.....Jeff Creamer

I have decided to try doing my bit to help dig up the best ideas for what the next edition of the Apple II family ought to be. I'm sending the "Wish-O-Gram" to everyone I can think of who might have some influence on a new Apple II.

In the first Wish-O-Gram I gave my opinions that the the next Apple II should be packed with hardware "horsepower"; be able to emulate the Mac and/or the IBM PC; have standard provisions for 20+ meg hard disk; have an empty socket for a math coprocessor, or one built in; have video output compatible with EGA/VGA monitors.

If any of you readers of Apple Assembly Line have some thoughts on the subject, I would sure like to hear them. Send them to Jeff Creamer, Electronics Technology, Yavapai College, Prescott, AZ 86301.

EnterSoft:

Basic-like macros which make the complex simple. Don't re-write that multiplication routine for the hundredth time! Get EnterSoft instead! Do 8/16/32/64 bit Math/Input-Output/Graphics simply without all of the hassles. These routines are a must for the serious programmer who doesn't want to spend all of his/her time trying to re-invent the wheel. DOS 3.3 Version=$30.00, ProDos Version=$30.00, BOTH for only $50.00. GET YOURS TODAY.

A Shape Table Program:

For once! A shape table program which is logically organizied into its componet parts. Each section resides in its own program. The editor, disk access, Hi-Res section; each section is separate. Written almost entirely in Basic, it is easily modified. Not copyprotected! Put them on a Hard Disk, Ram Drive, anywhere! DOS 3.3 Version=$20.00, ProDos Version=$20.00, BOTH for $30.00!

Send Check or Money Order To:

Mark Manning
c/o Simulacron I/Baggy Game
P.O. Box 58598
Webster, TX 77598

ProDos Upgrade for DOS 3.3
EnterSoft Owners = $20.00

Thanks for the letters -
Keep Writing!

Apple Assembly Line (ISSN 0889-4302) was published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228, from October, 1980 through May, 1988. Phone (214) 324-2050. This is the final issue. Back issues are $1.80 each for Volumes 1-7, $2.40 each from later Volumes (plus postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)